Natural Music Composition Using Elman Networks

Patrick Verga ([Pat@email.com](mailto:Pat@email.com))

Jacob Struzik ([jacobstruzik@gmail.com](mailto:jacobstruzik@gmail.com))

Department of Computer Science

University of Massachusetts

Amherst, MA 01002

**ABSTRACT**

We describe a method of training a system of Elman Networks to compose a piece of music. Training data consists of different collections of songs in MIDI format. Using a Matlab toolbox, MIDI files are read in, separated by tracks, and converted to note vectors to train one network per track. Songs are then constructed by having each network create a track which is then combined with the other network’s created tracks.

**INTRODUCTION**

The generation of musical scores via advanced machine learning has been a great interest for the past few decades. Many individuals have used statistical, syntactic, and evolutionary approaches to get a better understanding of how music theory can be adapted. These proposals have shown to work to an extent, but do not seem to capture a global structure or a consistent harmony.

It was not until 1994 when Michael Mozer proposed his CONCERT network architecture that a new idea of chord processing and psychophysical constraints was introduced [5]. Mozer’s network not only attempted to generate music by learning melodies but tried to attain an overall coherence of musical compositions. As opposed to previous methods, where algorithms would use transition tables to produce note-by-note compositions, Mozer’s network strives to predict classes of melodies by taking psychoacoustic attributes into account. Although the CONCERT network was complex in nature, it failed to represent global coherency of the desired genres of songs. Similar to Mozer’s paper, our experiment utilizes a recurrent neural network to classify melodies and interpret several psychophysical aspects of musical compositions such as duration, pitch, and instrument type.

Since Mozer’s paper, many other experiments involving deep learning and genetic variation [7] have been projected to artificially create music. Recently, Chen and Miikkulainen [1].

**METHODS**

**Network Architecture**

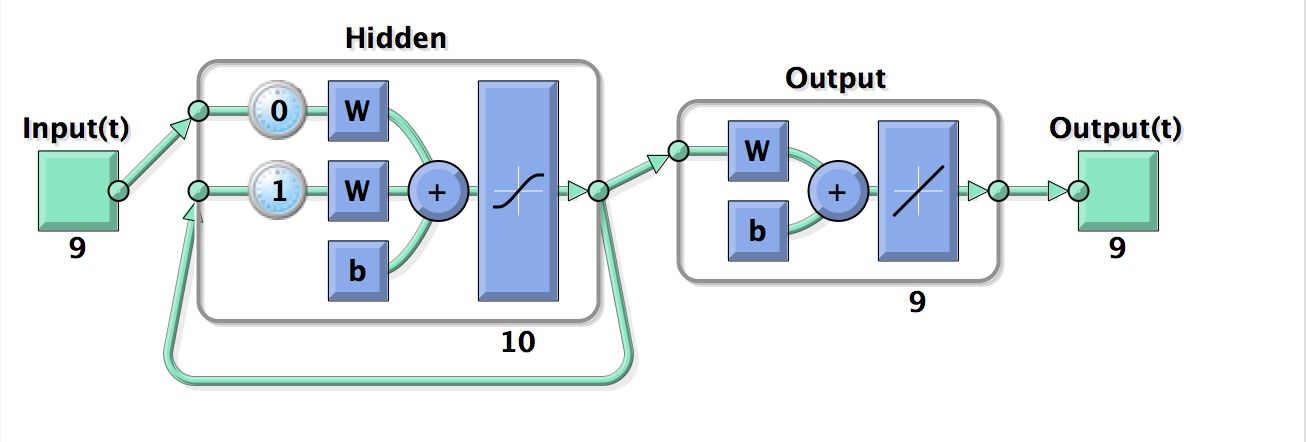
**Midi**

Our training input and song outputs are both in the form of Midi sound files. Midi files are a standardized representation of sound information meant to be interpreted and played by a digital synthesizer. The midi file contains information on both timings and pitch of notes played. Additionally, notes are separated into up to 16 tracks, each of which can have 1 of 16 instrument channels. Midi analysis was performed in Matlab using Ken Shutte’s Midi Toolbox (http://www.kenschutte.com/midi).

**Data Representation**

When we read in a midi file to be used as training data, we first separate the data into 16 different tracks based on the information in the file. Each track data is then sent to the corresponding track network. Each note is then converted to a 9 dimensional input vector. X0 represents the duration of the note, which is extracted from the Midi file. X9 is the instrument channel to be used. Our system uses one channel per track. The channel is chosen by the most occurring instrument channel in the corresponding track. X1 - X8 represent the first through 8th scale degree of a major scale (all of our songs are restricted to the major scale). If X1 = 1, then the current note is the first degree of the scale. All other X’s = 0 as only 1 note is chosen for each input vector.

**Elman Networks**



**(Figure 1: diagram of an Elman network)**

Our architecture uses a system of Simple Recurrent Neural Networks, aka Elman Networks (Elman, 1991), to create unique natural musical compositions. In a standard Elman network setup, a context layer is held within the network in addition to the input, output, and hidden layers. The context layer is used to retain information from the past and integrate it into the current actions of the network. Our system trains 16 different Elman networks, each corresponding to the 16 different tracks of the midi file. The training data is used to create input and output mappings. The input consists of each of the notes of the given track from each of the training Midi files in their vector form. The output mappings are created by shifting the input data by 1 note such that the input for note i of the song maps to the output of note i+1.

**Song Composition**

**Choosing Key**

A major component of a musical piece is the harmony which is defined predominently by the chord progression of the song. Our system allows for a dynamic harmony by the automated choosing and transitioning through different keys. In a normal song, it is easy to examine one or several bars of the song to discern the current key or chord. Because of the nature of Midi files, having many tracks and many notes being played simultaneously and in various rhythms, this is much harder.

To choose a current key we examine a window of notes around the current note of interest. We create a counter for each of the 12 possible keys. For each note within the window, if the note is the 1st, 3rd, or 5th (major chord) scale degrees of any of the keys, we add to the counter. Whichever key has the highest count after all notes are examined is chosen.

**Choosing Next Note**

The song creation process is begun by constructing a vector corresponding to an arbitrary note which is fed into each of the track networks. Each network then outputs a vector of note duration and scale degree probabilities. We perform a weighted roulette on the probability distribution to choose exactly one note degree for the vector. This note vector is then fed back into the network to create the next note.

**Rhythm Timing**

A consistent rhythm tying together the various aspects of a musical piece is a crucial component in most musical genres. The rhythm section of the band, typically consisting of drums and bass, usually performs this task. We attempt to replicate the aspect of rhythm by finding a steady beat to play new notes on. We find the natural timing of the training data by looking at the note durations of track 10 of our Midi training data, if it exists. This is because track 10 is traditionally reserved for a rhythm instrument. If track 10 does not exist, we use the average note durations over all tracks.

**Output Generation**

We define a variable T to denote the total length of time for the song. For every beat, found above, we choose a note vector for each track. The total number of notes per track will therefore be (total time / beat time). Each track’s note choices are then combined into a single Midi file.

**RESULTS**

For quantitative results we can discuss error rates when our network is classifying on a few of the different types of songs (simple to more complex).

For qualitative we can either get several people to listen to our tracks or analyze the tracks ourselves recognizing common music theory elements.

**(Figure 2: Note distributions for network output of notes chosen for Mario song output)**

**Song 1: Mario**

Memory:Users:pat:Desktop:prettygood-mario.pdf

**(Figure 3: Sheet music of song created using Mario training data. Generated using MidiSheetMusic (http://midisheetmusic.sourceforge.net/))**

**Song 2: Mozart**Memory:Users:pat:Desktop:mozart.pdf**(Figure 4: Sheet music for song composed on Mozart training generated using MidiSheetMusic (http://midisheetmusic.sourceforge.net/))**

**(Figure 5: Note distributions for network output)**

**Note Distribution Graphs**

Figures 2 and 5 show the average note distributions for each of the tracks within the Mario and mozart songs respectively. The note distribution is the output of the neural network that is used to choose the next note. Figure 2 Track 4 shows a steady drum beat which is why there is only a single note choice consistently.

Figures 3 and 4 show the sheet music versions of two created songs, one using Mozart and one using Mario training data. These were generated using MidiSheetMusic (<http://midisheetmusic.sourceforge.net/>)

**DISCUSSION**

Our system for natural autonomous musical composition is able to learn musical relations within training data and create its own songs. These compositions demonstrate musical properties such as melody, rhythm, and harmony and are generated autonomously.

Future directions for this research include having a tighter coupling between the different track networks. The current implementation treats each track independently, essentially stitching together small songs into a larger song. The global beat timing and key choice allows these different tracks to work together. A better scheme would be for each network to not only base note choices on their own past choices, but also on the note choices of the other network tracks.

At the same time, an option for independent rhythms rather than a single global one would make a more diverse and natural composition. Other future work includes trying different neural network architectures.

**REFERENCES**

[1] Chen, C-CJ, and Risto Miikkulainen. "Creating melodies with evolving recurrent neural networks." *Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on*. Vol. 3. IEEE, 2001.

[2] D’Annunzio, David, and Kenneth Ning. "Learning Classes of Melodies with a Recurrent Neural Network." (2012).

[3] Elman, Jeffrey L. "Distributed representations, simple recurrent networks, and grammatical structure." *Machine learning* 7.2 (1991): 195-225.

[4] Franklin, Judy A. "Recurrent neural networks for music computation."*INFORMS Journal on Computing* 18.3 (2006): 321-338.

[5] MICHAEL, C. MOZER. "Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing." *Connection Science* 6.2-3 (1994): 247-280.

[6] Sergent, Justine, et al. "Distributed neural network underlying musical sight-reading and keyboard performance." *Science* (1992).

[7] Tokui, Nao, and Hitoshi Iba. "Music composition with interactive evolutionary computation." *Proceedings of the 3rd international conference on generative art*. Vol. 17. No. 2. 2000.